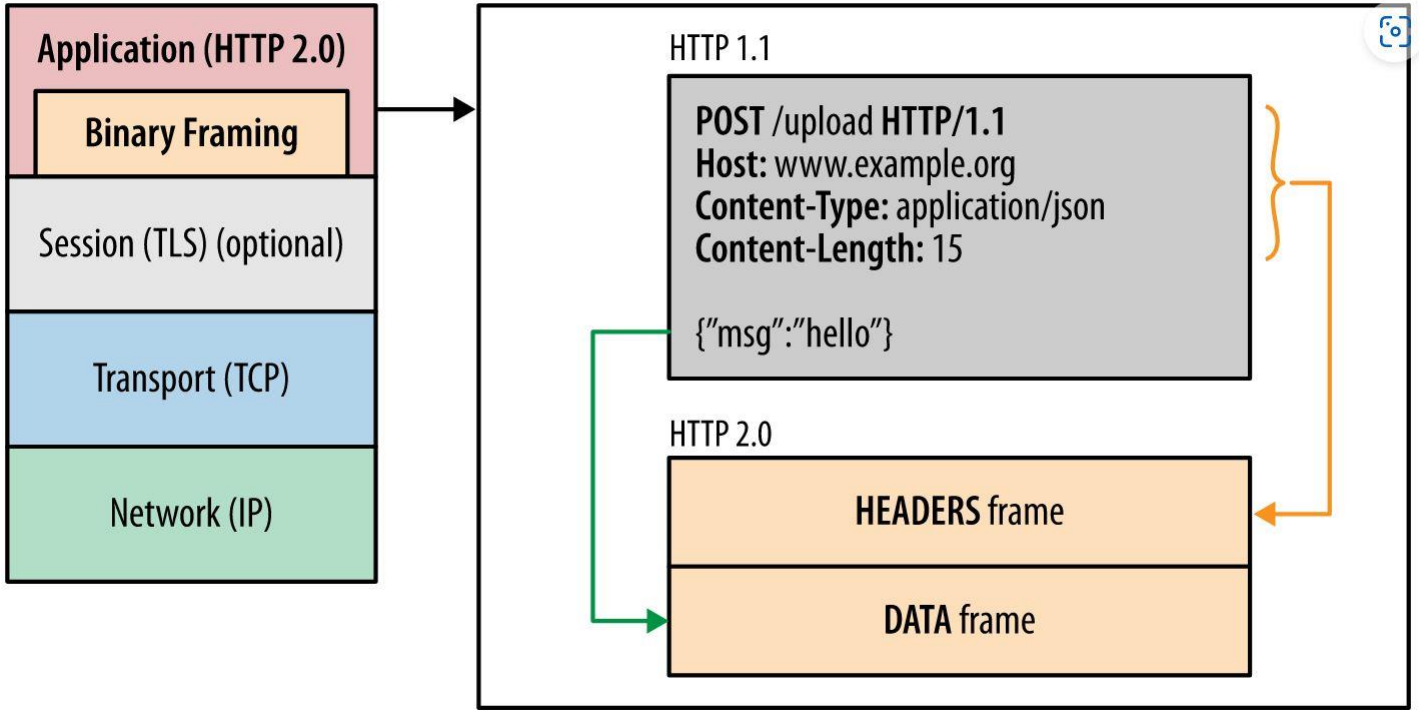
Important Features of HTTP/2 (and HTTP/3)

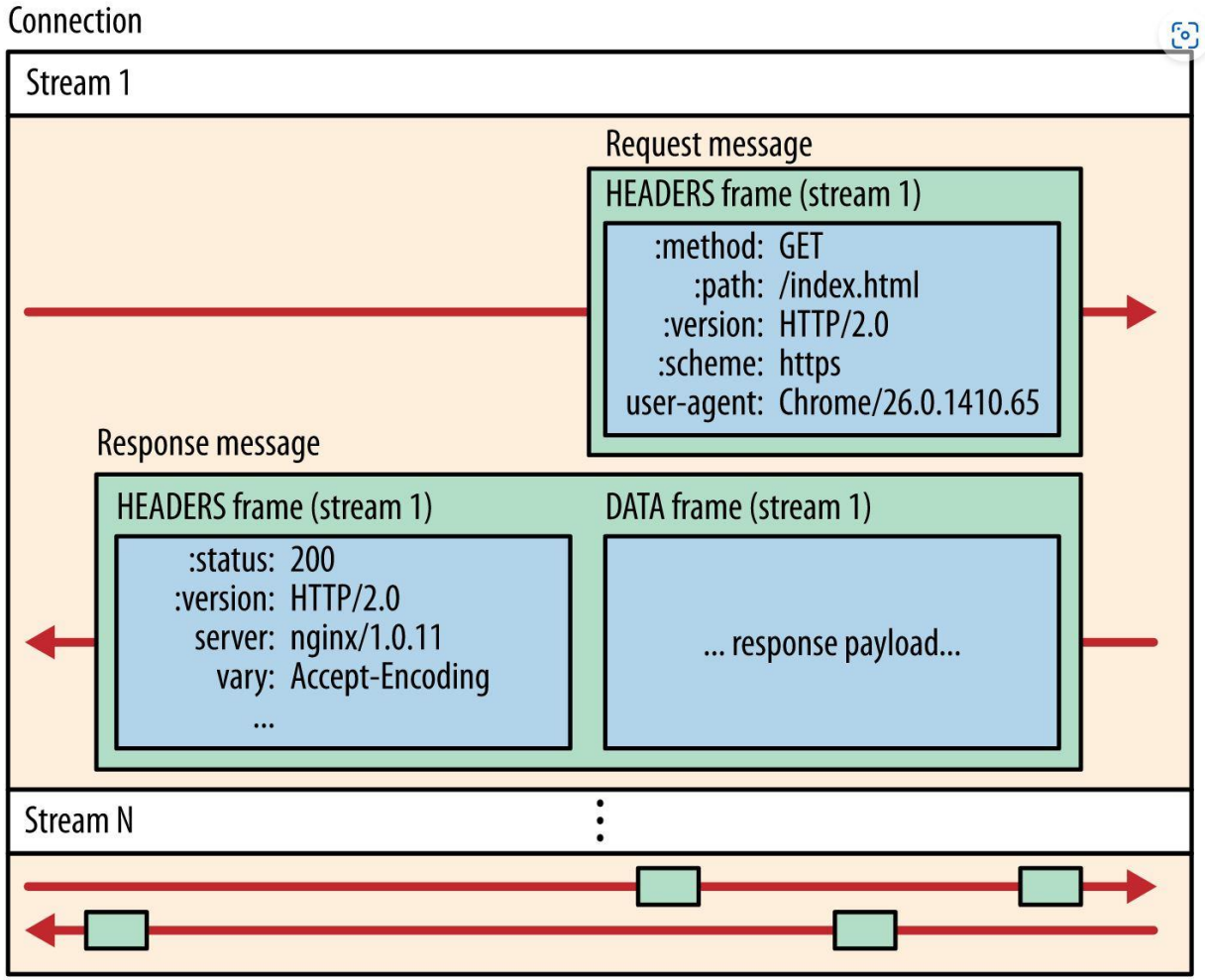
HTTP/2 Features

1. Binary framing Layer
2. Request and response (stream) multiplexing
3. Header compression (HPACK)
4. Server push
5. Stream priority

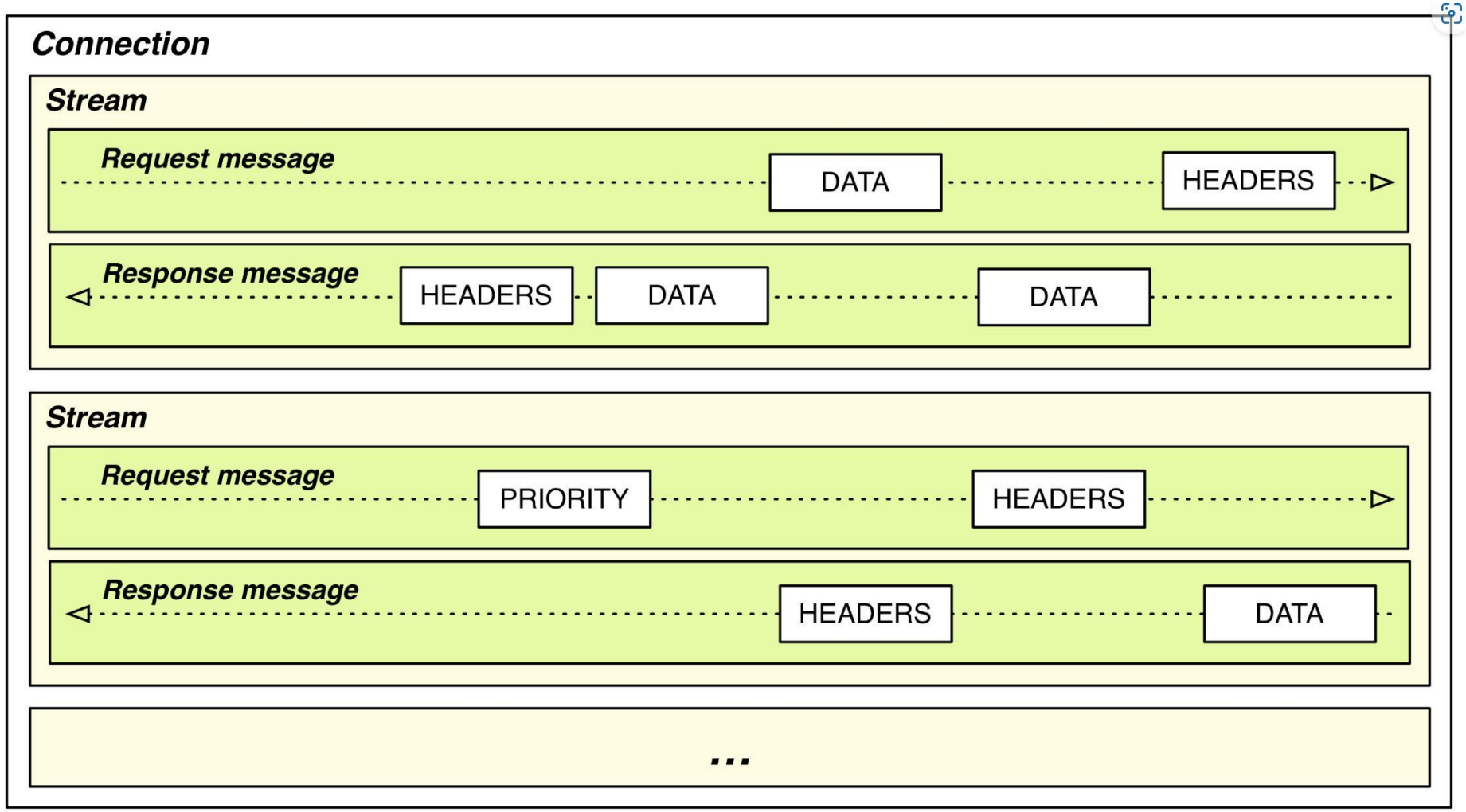
Binary Frame Header



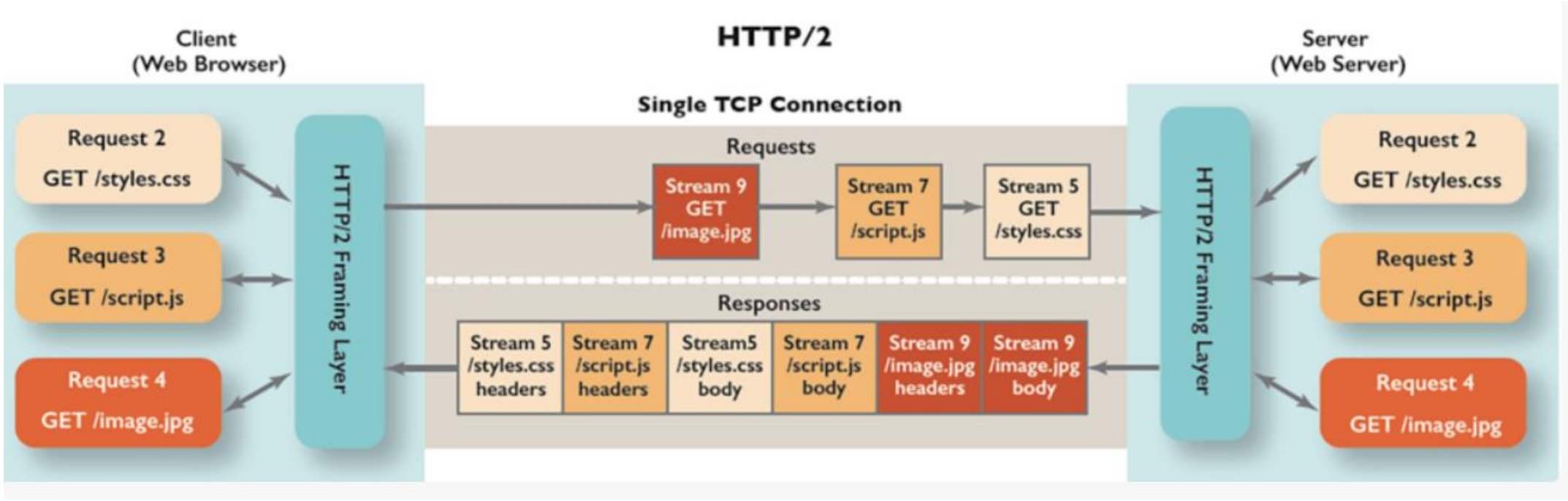
Binary Frame Header (Cont'd)



Binary Frame Header (Cont'd)



Request and Response (Stream) Multiplexing



Frame Format

4.1. Frame Format

All frames begin with a fixed 9-octet header followed by a variable-length payload.

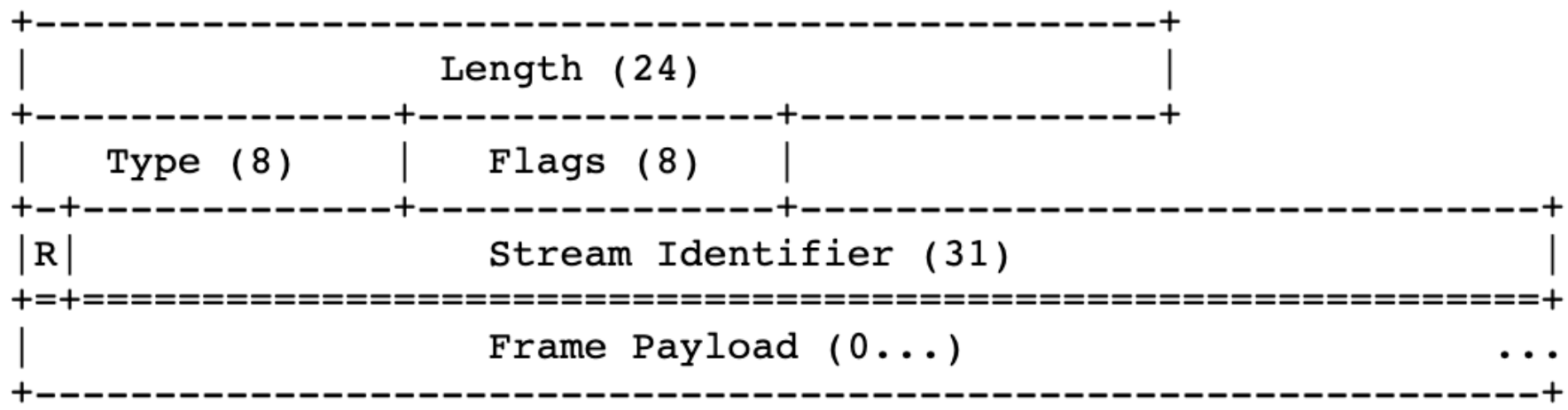


Figure 1: Frame Layout

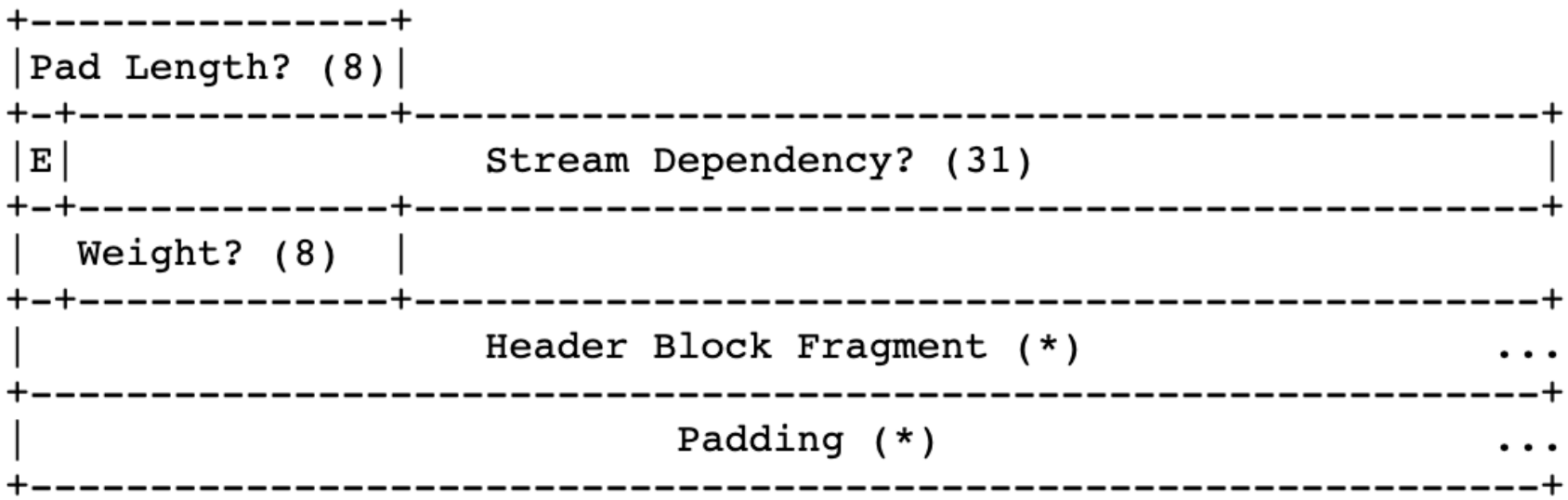
Supported Types in the Frame

- DATA
  - Used to transport the body of a HTTP message
- HEADERS
  - Used to communicate the header fields of a HTTP message
- PRIORITY
  - Used to communicate sender-advised priority of a stream
- RST_STREAM
  - Used to signal termination of a stream
- SETTINGS
  - Used to communicate configuration parameters for the connection

Supported Types in the Frame (cont’d)

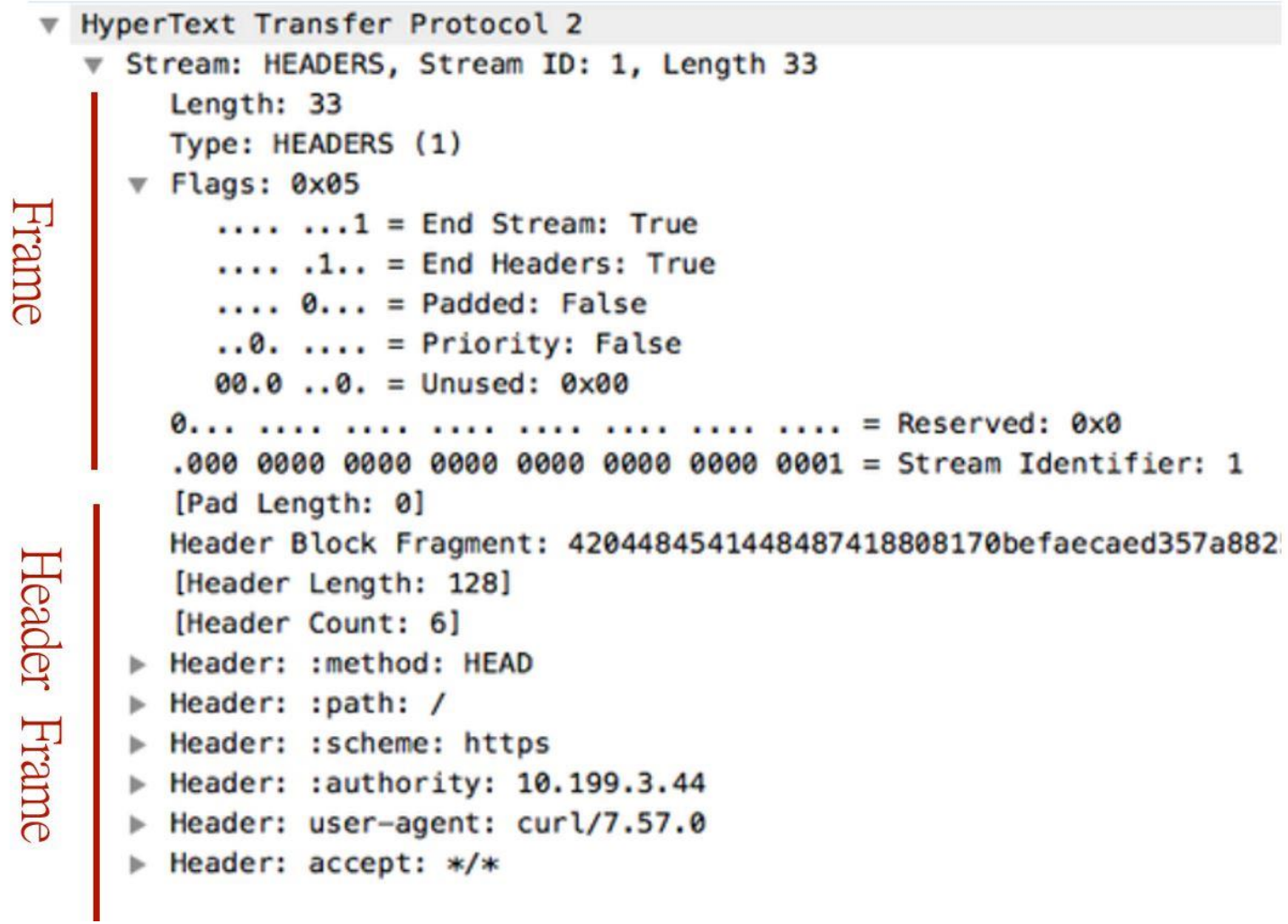
- PUSH_PROMISE
  - Used to signal a promise to serve the referenced resource. The server will actively push the resources to the client (to reduce latency).
- CONTINUATION
  - Used to continue a sequence of header block fragments
- PING
  - Used to measure the roundtrip time and perform "liveness" checks
- WINDOW_UPDATE
  - Used to implement flow stream and connection flow control
- GOAWAY
  - Used to inform the peer to stop creating streams for current connection

Header Frame

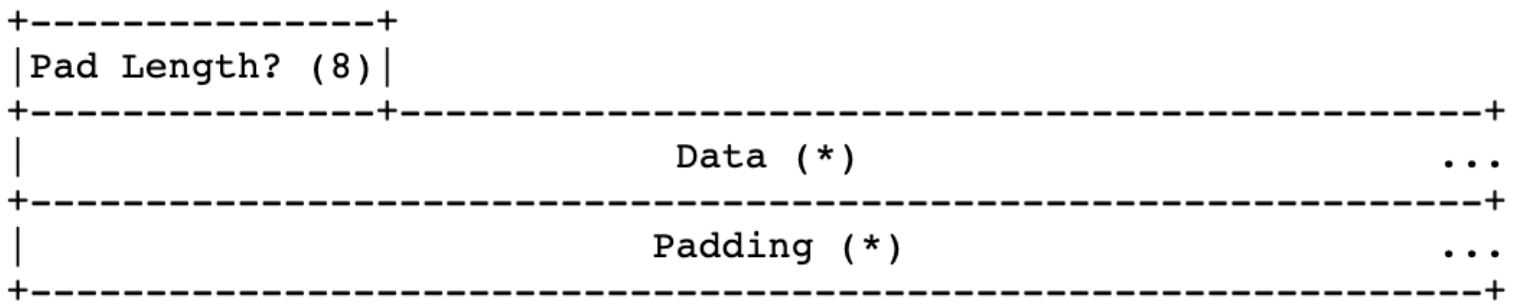


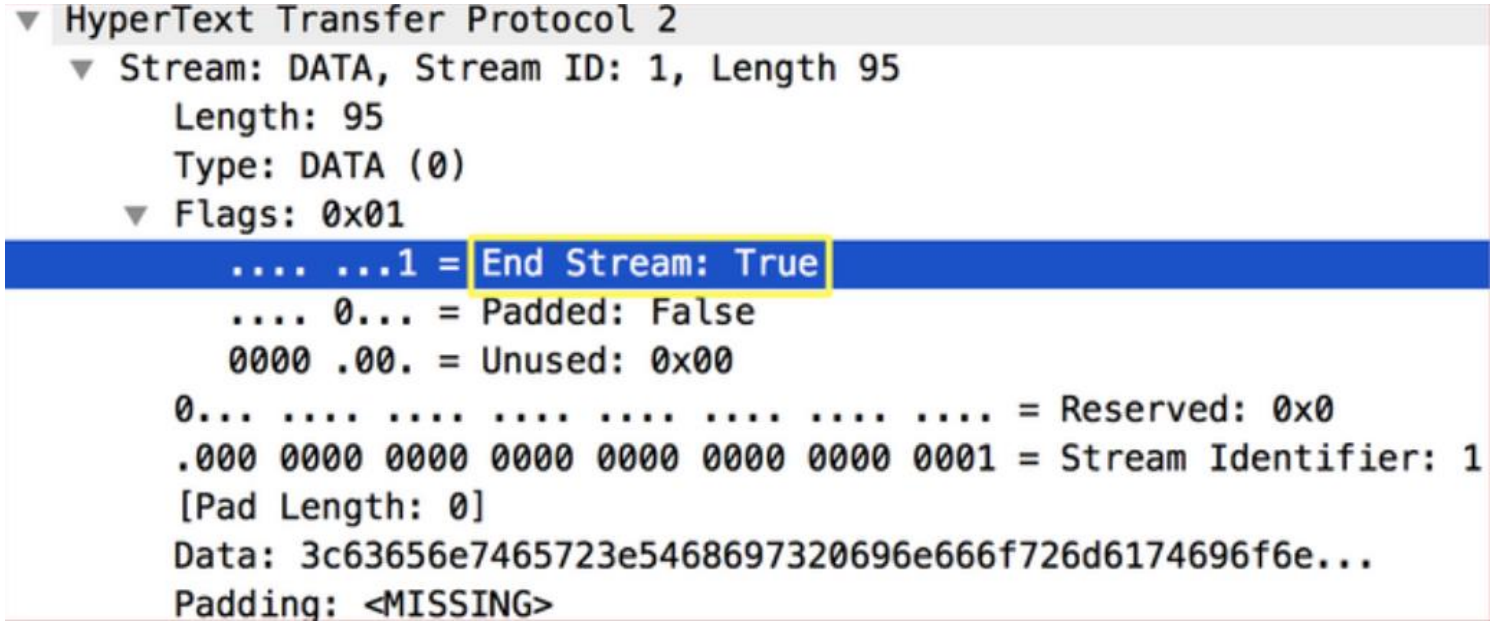
- Padding is a security feature.
- The header block is divided into one or more header block fragments, and then transmitted within the payload of a HEADERS, PUSH_PROMISE, or CONTINUATION frame.

Header Frame (cont'd)

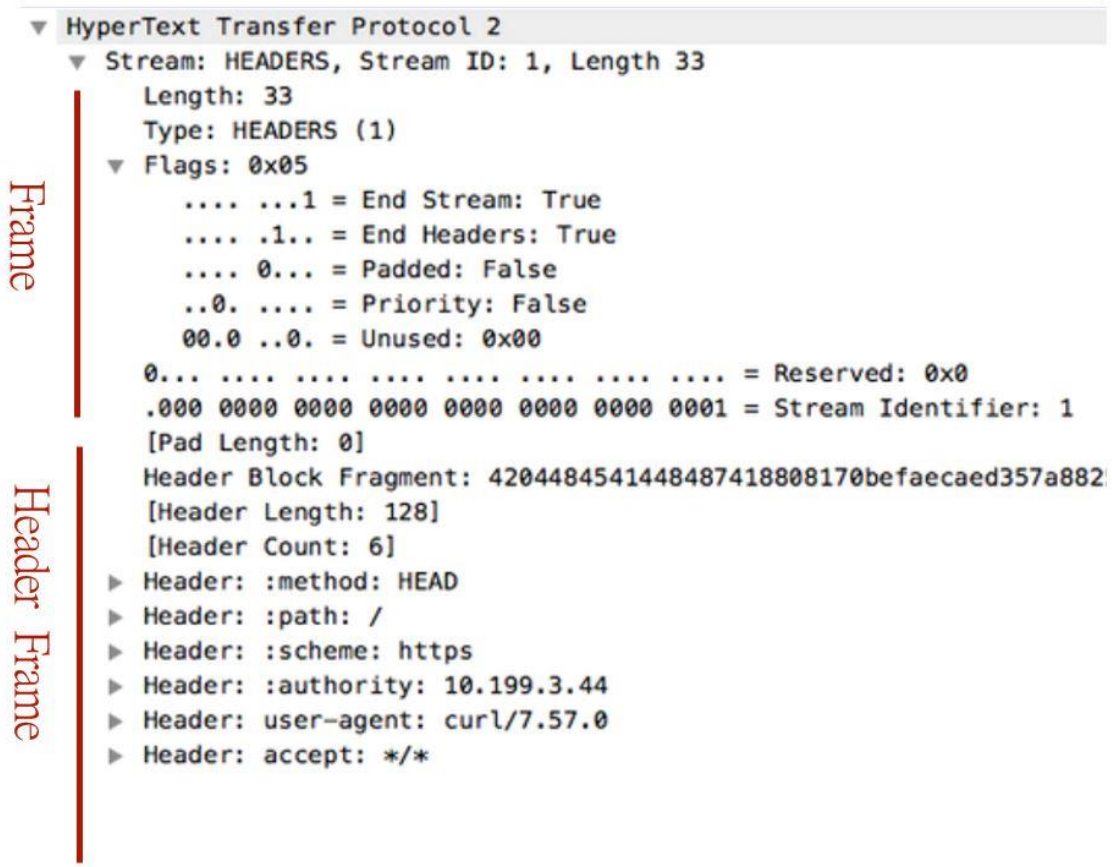


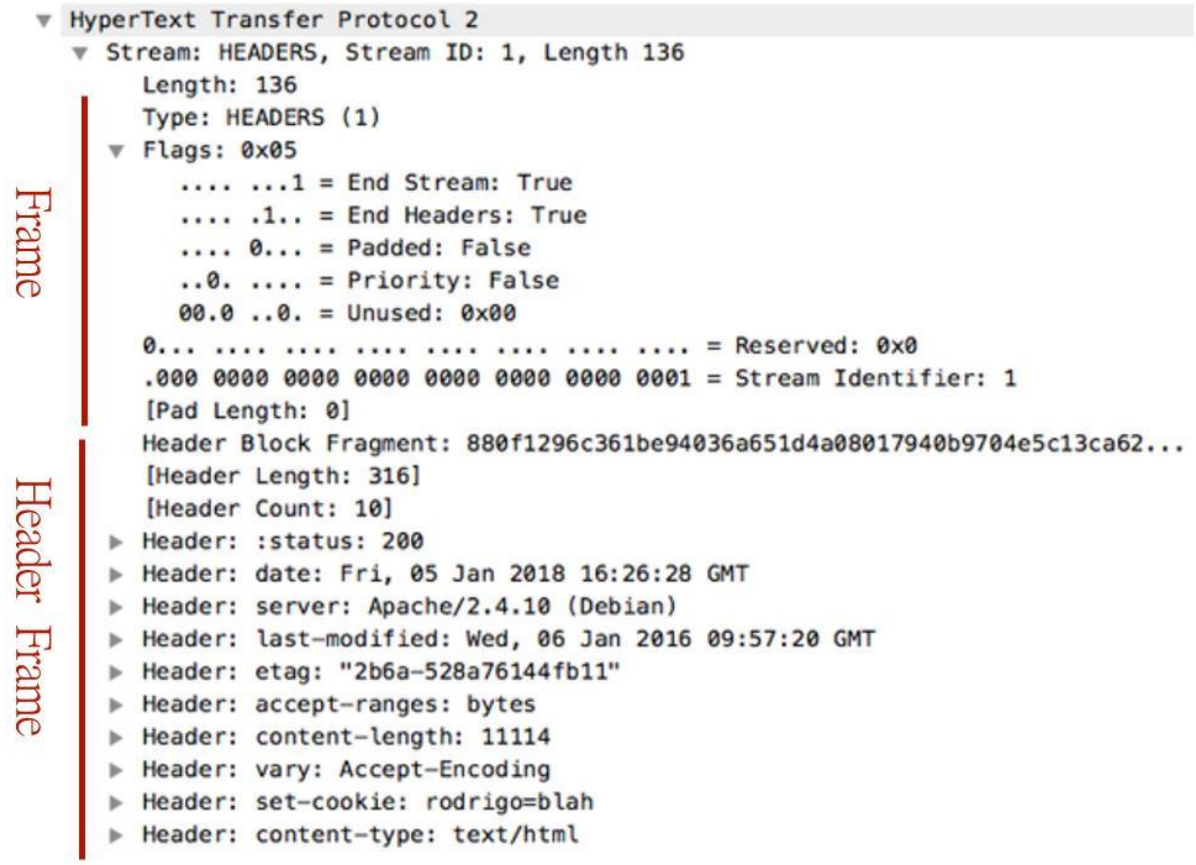
Data Frame


+---------------+
|Pad Length? (8)|
+---------------+---+
| Data (*)   ...
+---+
| Padding (*)   ...
+---+

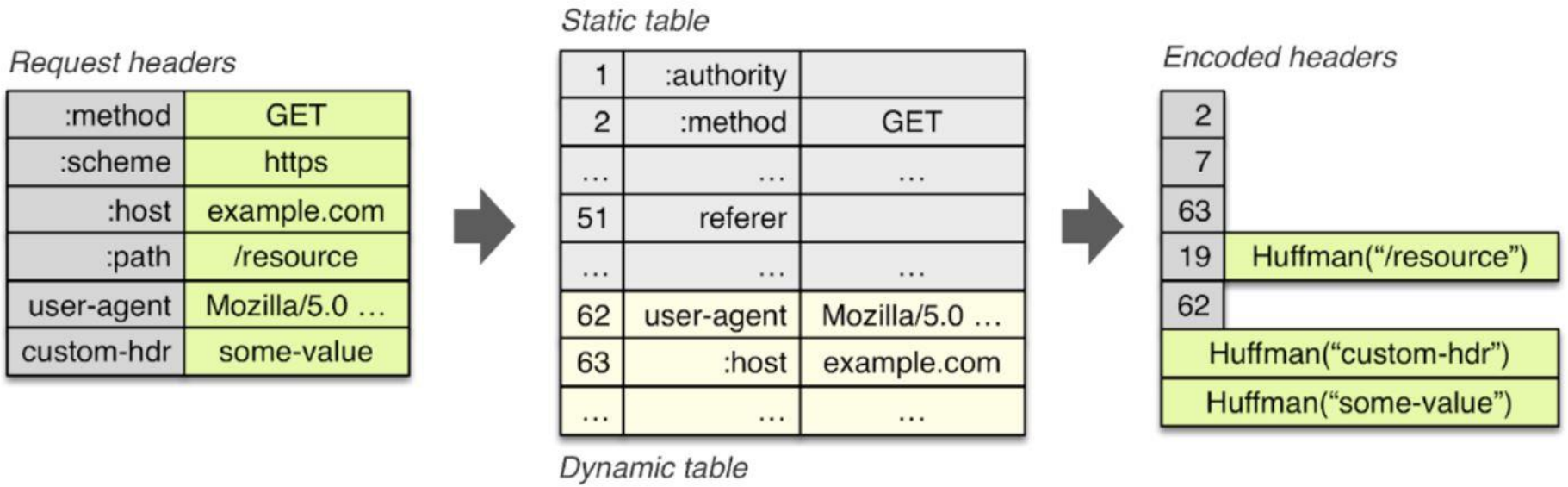

HyperText Transfer Protocol 2
Stream: DATA, Stream ID: 1, Length 95
Length: 95
Type: DATA (0)
Flags: 0x01
....1 = End Stream: True
.... 0... = Padded: False
0000 .00. = Unused: 0x00
0... .... = Reserved: 0x0
.000 0000 0000 0000 0000 0000 0000 0001 = Stream Identifier: 1
[Pad Length: 0]
Data: 3c63656e7465723e5468697320696e666f726d6174696f6e...
Padding: <MISSING>

HTTP Request and Response





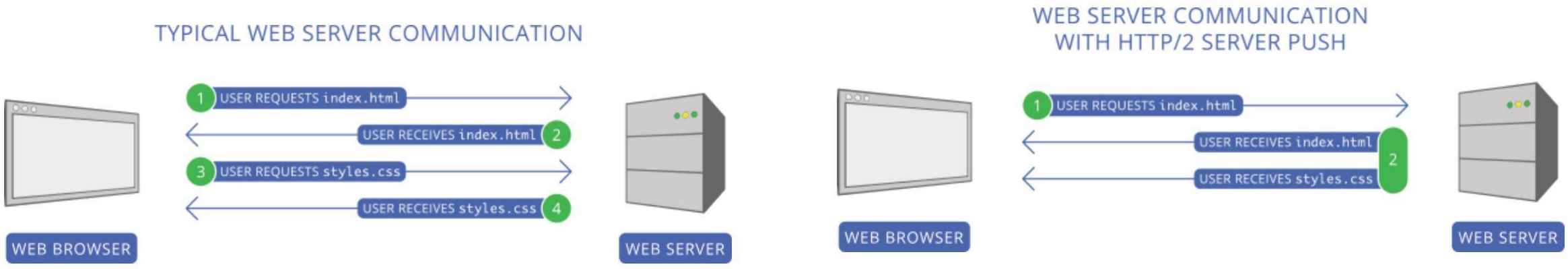
Header Compression (HPACK)



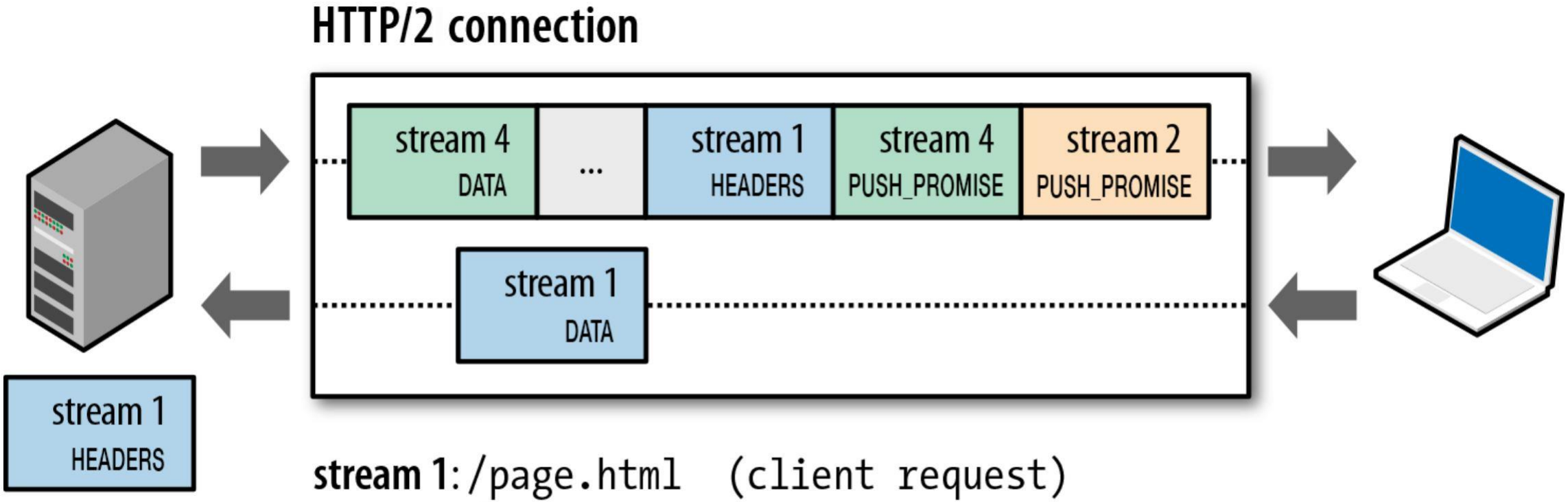
- *Literal values are (optionally) encoded with a static Huffman code*
- *Previously sent values are (optionally) indexed*
  - *e.g. "2" in above example expands to "method: GET"*

Server Push

- Server push allows you to send server-site resources to the clients before they've even asked for them.
- This can reduce the latency of fetching these resources.



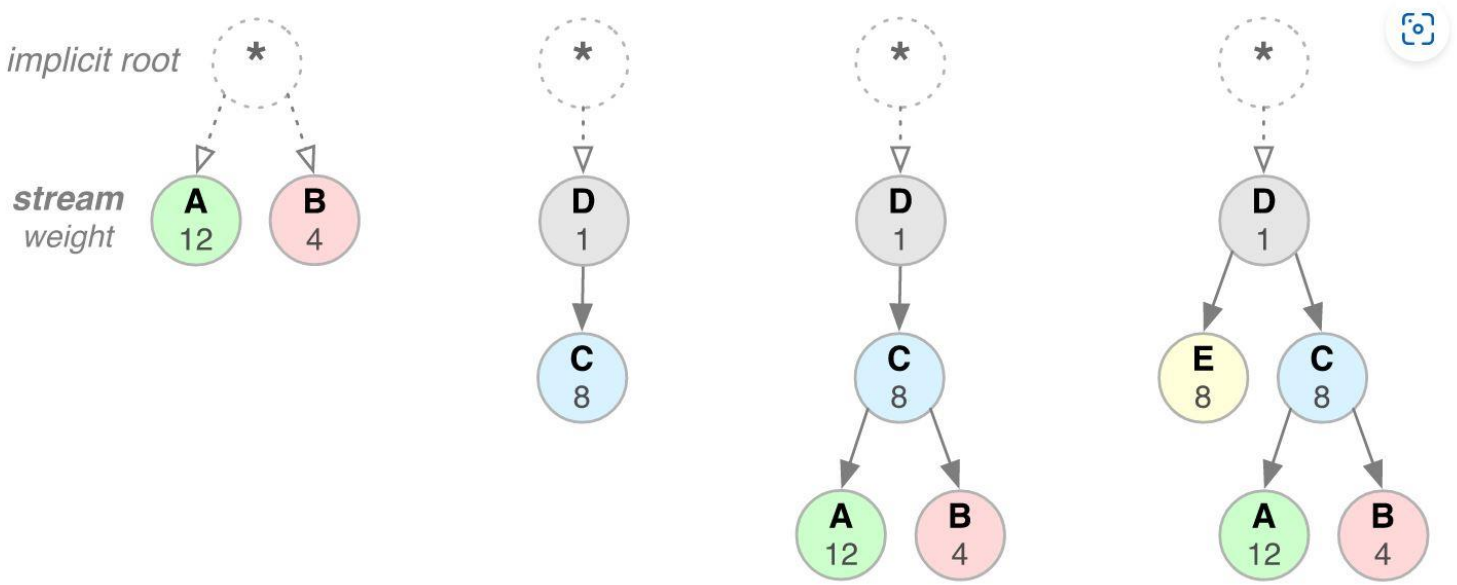
Server Push (cont'd)



stream 1: /page.html (client request)
stream 2: /script.js (push promise)
stream 4: /style.css (push promise)

Stream Priority

- A stream dependency within HTTP/2 is declared by referencing the unique identifier of another stream as its parent.
- If omitted, the stream is said to be dependent on the “root stream”.
- Declaring a stream dependency indicates that, if possible, the parent stream should be allocated resources ahead of its dependencies



Relationships between HTTP/2, HTTP/3 and QUIC

- HTTP/2 was standardized as RFC in 2015. At that time, QUIC has not been proposed.
- In 2021, QUIC was standardized as RFC to solve the TCP HOL blocking problem with HTTP/2.
- In 2022, HTTP/3 was standardized as RFC to work with QUIC.
- HTTP/3 is a slimmed version of HTTP/2. Most functions of HTTP/2 and HTTP/3 are the same.
- HTTP/2 (and thus HTTP/3) and QUIC protocols both support stream multiplexing over a connection.
- There is a one-by-one stream mapping between HTTP/3 and QUIC.